Employing *DL-Lite*$_R$-Reasoners for Fuzzy Query Answering

Theofilos Mailis(✉) and Anni-Yasmin Turhan

Chair for Automata Theory, Theoretical Computer Science,
TU Dresden, Dresden, Germany
mailis@tcs.inf.tu-dresden.de

Abstract. Fuzzy Description Logics generalize crisp ones by providing membership degree semantics for concepts and roles by fuzzy sets. Recently, answering of conjunctive queries has been investigated and implemented in optimized reasoner systems based on the rewriting approach for crisp DLs. In this paper we investigate how to employ such existing implementations for crisp query answering in *DL-Lite*$_R$ over *fuzzy* ontologies. To this end we give an extended rewriting algorithm for the case of fuzzy *DL-Lite*$_R$-ABoxes that employs the one for crisp *DL-Lite*$_R$ and investigate the limitations of this approach. We also tested the performance of our proto-type implementation FLite of this method.

1 Introduction

Description Logics (DLs) are a class of knowledge representation languages with well-defined semantics that are widely used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. Some applications require to describe sets for which there exists no sharp, unambiguous distinction between the members and nonmembers. For example, when classifying numerical sensor values into symbolic classes, a crisp (non-fuzzy), unambiguous distinction between the members and nonmembers is not a natural way of modeling. To represent this kind of information faithfully, fuzzy variants of DLs were introduced. These variants generalize crisp ones by providing membership degree semantics for their concepts and roles by fuzzy sets.

In the last years conjunctive query answering was the main reasoning task investigated for DLs. This reasoning task allows to access data in a flexible way. In order to cope with huge amounts of data, the property of first order (FOL) rewritability of DLs was defined and investigated. This property of a DL allows to implement query answering by a two step procedure: First, the initial query is rewritten such that it captures the information from the TBox. Second, this query is executed over a database capturing the facts from the ABox by means of SQL queries. FOL rewritability is the key feature of the DL-Lite family, which has been proposed and investigated in [2]. It guarantees that query answering can be done efficiently—in the size of the data and in the overall size

Partially supported by DFG SFB 912 (HAEC).

T. Supnithi et al.(Eds.): JIST 2014, LNCS 8943, pp. 63–78, 2015.
DOI: 10.1007/978-3-319-15615-6_5

of the corresponding ontology. This is the main reason why *DL-Lite*$_R$ is the DL underlying OWL 2 QL, one of the three profiles of OWL 2 language.

So far several fuzzy extensions of DL-Lite have been investigated. In [11,12] the problem of evaluating ranked top-k queries in fuzzy DL-Lite is considered, and a variety of query languages by which a fuzzy DL-Lite knowledge base can be queried is presented in [6]. Though all of these approaches are tractable w.r.t. data complexity, they do not exploit the optimized query rewriting techniques that have been implemented in many systems for the classical case such as QuOnto2 [1,7], Ontop [8], Owlgres [9], and IQAROS [14]. There are also reduction techniques for very expressive DLs such as $\mathcal{SHIQ}$ from fuzzy to crisp [5] for query answering. These techniques are not promising in terms of efficiency, since they don't allow for FOL rewriting-based algorithms that employ relational databases (as the DL-Lite family).

Our approach to answering conjunctive queries over fuzzy *DL-Lite*$_R$-ontologies is to use existing optimized crisp DL-Lite reasoners as a black box to obtain an initial rewriting of the conjunctive query. We extend this query by (1) fuzzy atoms and (2) by so-called degree variables that capture the numerical membership degrees, which are used to return the corresponding fuzzy degrees. This straightforward approach allows to employ a standard SQL query engine—as in the crisp case and is thus easy to implement. It gives correct answers for the Gödel family of operators, which is widely used. However, for other families of fuzzy operators, answers concerning the degrees may be incorrect. We give a characterization of such cases and an estimation function for the interval in which the correct degrees lie. We have implemented the query answering engine FLite based on this approach for fuzzy *DL-Lite*$_R$, which uses the Ontop system [8] to obtain the initial crisp rewriting.

The rest of the paper is structured as follows: next, we introduce fuzzy *DL-Lite*$_R$. Section 3 presents the algorithms for consistency checking and query answering for fuzzy *DL-Lite*$_R$-ontologies. In Section 4 we describe limitations of our approach: we characterize the cases in which incorrect results are obtained and why other fuzzy extensions of DL-Lite-ontologies are problematic. The FLite system, based on the Ontop framework, is described and evaluated in Section 5. We end with conclusions and future work.

2 Preliminaries

We introduce the logic *DL-Lite*$_R$, its ontologies and then the fuzzy variant of the latter [6,12]. Starting from a set of concept names $\mathsf{N_C}$ and role names $\mathsf{N_R}$ complex concepts can be build. *DL-Lite*$_R$ distinguishes basic concepts represented by B, general concepts represented by C, basic roles represented by Q, and general roles represented by R, by the grammar:

$$B \rightarrow A \mid \exists Q \qquad C \rightarrow \top \mid B \mid \neg B \qquad Q \rightarrow P \mid P^- \qquad R \rightarrow Q \mid \neg Q$$

where $\top$ is the top concept. A *degree* d is a number from the unit interval [0, 1]. The *DL-Lite*$_R$-concepts and -roles are used in axioms, which can have the

Table 1. Families of fuzzy logic operators

Family	t-norm $a \otimes b$	negation $\ominus a$	implication $\alpha \Rightarrow b$
Gödel	$\min(a, b)$	$\begin{cases} 1, & a = 0 \\ 0, & a > 0 \end{cases}$	$\begin{cases} 1, & a \leqslant b \\ b, & a > b \end{cases}$
Łukasiewicz	$\max(a + b - 1, 0)$	$1 - a$	$\min(1 - a + b, 1)$
Product	$a \times b$	$\begin{cases} 1, & a = 0 \\ 0, & a > 0 \end{cases}$	$\begin{cases} 1, & a \leqslant b \\ b/a, & a > b \end{cases}$

following forms:

$$\begin{aligned} B \sqsubseteq C \quad & \text{(general concept inclusion axiom)} \\ Q \sqsubseteq R \quad & \text{(role inclusion axiom)} \\ \text{funct}(Q) \quad & \text{(functionality axiom)} \end{aligned}$$

A TBox $\mathcal{T}$ is a finite set of axioms. The set $\mathsf{N_I}$ is the set of individual names. Let $a, b \in \mathsf{N_I}$ and d be a degree, then a *fuzzy assertion* is a statement of the form:

$$\begin{aligned} \langle B(a), d \rangle \quad & \text{(fuzzy concept assertion)} \\ \langle P(a, b), d \rangle \quad & \text{(fuzzy role assertion)} \end{aligned}$$

An ABox $\mathcal{A}$ is a finite set of fuzzy assertions. A fuzzy DL-Lite ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The crisp *DL-Lite*$_R$-ontologies (ABoxes) are a special case, where only degrees $d = 1$ are admitted.

The semantics of fuzzy *DL-Lite*$_R$ are provided via the different families of fuzzy logic operators depicted in Table 1 and interpretations. An *interpretation* for fuzzy *DL-Lite*$_R$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is the interpretation domain and $\cdot^{\mathcal{I}}$ is an interpretation function mapping every individual a onto an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, every concept name A onto a *concept membership function* $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$, every atomic role P onto a *role membership function* $P^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$.

Let δ, δ' denote elements of $\Delta^{\mathcal{I}}$ and $\ominus$ denote fuzzy negation (Table 1), then the semantics of concepts and roles are inductively defined as follows:

$$(\exists Q)^{\mathcal{I}}(\delta) = \sup_{\delta' \in \Delta^{\mathcal{I}}} Q^{\mathcal{I}}(\delta, \delta') \qquad (\neg B)^{\mathcal{I}}(\delta) = \ominus B^{\mathcal{I}}(\delta) \qquad \top^{\mathcal{I}}(\delta) = 1$$

$$P^{-\mathcal{I}}(\delta, \delta') = P^{\mathcal{I}}(\delta', \delta) \qquad (\neg Q)^{\mathcal{I}}(\delta, \delta') = \ominus Q^{\mathcal{I}}(\delta, \delta')$$

We say an interpretation $\mathcal{I}$ *satisfies* a

- concept inclusion axiom $B \sqsubseteq C$ iff $B^{\mathcal{I}}(\delta) \leqslant C^{\mathcal{I}}(\delta)$ for every $\delta \in \Delta^{\mathcal{I}}$,
- role inclusion axiom $Q \sqsubseteq R$ iff $Q^{\mathcal{I}}(\delta, \delta') \leqslant R^{\mathcal{I}}(\delta, \delta')$ for every $\delta, \delta' \in \Delta^{\mathcal{I}}$,
- functionality axiom func(Q) iff for every $\delta \in \Delta^{\mathcal{I}}$ there is a unique $\delta' \in \Delta^{\mathcal{I}}$ such that $Q^{\mathcal{I}}(\delta, \delta') > 0$.

We say that an interpretation $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$, i.e. $\mathcal{I} \models \mathcal{T}$, iff it satisfies all axioms in $\mathcal{T}$. $\mathcal{I}$ satisfies a fuzzy concept assertion $\langle B(a), d \rangle$ iff $B^{\mathcal{I}}(a^{\mathcal{I}}) \geqslant d$,

and a fuzzy role assertion $\langle P(a,b), d\rangle$ iff $P^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \geqslant d$. $\mathcal{I}$ is a *model of an ABox* $\mathcal{A}$, i.e. $\mathcal{I} \models \mathcal{A}$, iff it satisfies all assertions in $\mathcal{A}$. Finally an interpretation $\mathcal{I}$ is a model of an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ iff it is a model of $\mathcal{A}$ and $\mathcal{T}$.

Based on the formal semantics several reasoning problems can be defined for DLs. A *DL-Lite*$_R$-concept or TBox is *satisfiable* iff it has a model. Likewise, a *DL-Lite*$_R$-ontology is *consistent* iff it has a model, otherwise it is *inconsistent*. Given a TBox $\mathcal{T}$ and two concepts C and D, C is *subsumed by* D w.r.t. $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$), iff for all models $\mathcal{I}$ of $\mathcal{T}$ $C^{\mathcal{I}}(\delta) \leq D^{\mathcal{I}}(\delta)$ holds. The reasoning problem we want to address in this paper is answering of (unions of) conjunctive queries, which allows retrieval of tuples of individuals from the ontology by the use of variables. Let $\mathsf{N_V}$ be a set of variable names and let $t_1, t_2 \in \mathsf{N_I} \cup \mathsf{N_V}$ be terms (either individuals or variable names). An *atom* is an expression of the form: $C(t_1)$ (*concept atom*) or $P(t_1, t_2)$ (*role atom*). Let $\mathbf{x}$ and $\mathbf{y}$ be vectors over $\mathsf{N_V}$, then $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms of the form $A(t_1)$ and $P(t_1, t_2)$. A *conjunctive query* (CQ) $q(\mathbf{x})$ over an ontology $\mathcal{O}$ is a first-order formula $\exists \mathbf{y}.\phi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ are the *answer variables*, $\mathbf{y}$ are *existentially quantified variables* and the concepts and roles in $\phi(\mathbf{x}, \mathbf{y})$ appear in $\mathcal{O}$. Observe, that the atoms in a CQ do not contain degrees. A *union of conjunctive queries* (UCQ) is a finite set of conjunctive queries that have the same number of answer variables.

Given a CQ $q(\mathbf{x}) = \exists \mathbf{y}.\phi(\mathbf{x}, \mathbf{y})$, an interpretation $\mathcal{I}$, a vector of individuals $\boldsymbol{\alpha}$ with the same arity as $\mathbf{x}$, we define the mapping π that maps: i) each individual a to $a^{\mathcal{I}}$, ii) each variable in $\mathbf{x}$ to a corresponding element of $\boldsymbol{\alpha}^{\mathcal{I}}$, and iii) each variable in $\mathbf{y}$ to a corresponding element $\delta \in \Delta^{\mathcal{I}}$. Suppose that for an interpretation $\mathcal{I}$, Π is the *set of mappings* that comply to these three conditions. Computing the t-norm $\otimes$ of all atoms: $A^{\mathcal{I}}(\pi(t_1))$ and $P^{\mathcal{I}}(\pi(t_1), \pi(t_2))$ yields the degree of $\phi^{\mathcal{I}}(\boldsymbol{\alpha}^{\mathcal{I}}, \pi(\mathbf{y}))$. A tuple of individuals $\boldsymbol{\alpha}$ is a *certain answer* to $q(\mathbf{x})$, over $\mathcal{O}$, with a degree greater or equal than d (denoted $\mathcal{O} \models q(\boldsymbol{\alpha}) \geqslant d$), if for every model $\mathcal{I}$ of $\mathcal{O}$:

$$q^{\mathcal{I}}(\boldsymbol{\alpha}^{\mathcal{I}}) = \sup_{\pi \in \Pi} \{\phi^{\mathcal{I}}(\boldsymbol{\alpha}, \pi(\mathbf{y}))\} \geqslant d.$$

We denote the set of certain answers along with degrees, to a query $q(\mathbf{x})$ w.r.t. an ontology $\mathcal{O}$ with $ans(q(\mathbf{x}), \mathcal{O})$:

$$ans(q(\mathbf{x}), \mathcal{O}) = \{(\boldsymbol{\alpha}, d) \mid \mathcal{O} \models q(\boldsymbol{\alpha}) \geqslant d \text{ and there exists no } d' > d \text{ such that } \mathcal{O} \models q(\boldsymbol{\alpha}) \geqslant d'\}.$$

A special case of CQs and UCQs are those with an empty vector $\mathbf{x}$ of answer variables. These queries return only a degree of satisfaction and are called *degree queries*. An ontology *entails a degree query* q to a degree of d, if $\mathcal{O} \models q() \geqslant d$ and $\mathcal{O} \not\models q() \geqslant d'$ for every $d' > d$. In the crisp case, these queries are *Boolean queries* and return true of false. A crisp ontology entails a Boolean query q, if $\mathcal{O} \models q()$.

Example 1. To illustrate the expressiveness of fuzzy *DL-Lite*$_R$, we give an example from the operating systems domain focusing on information about servers.

The first two concept inclusions in the TBox $\mathcal{T}_{ex}$ state that each server has a part that is a CPU. The functional restriction states that no CPU can belong to more than one server. The ABox $\mathcal{A}_{ex}$ provides information about the connections between servers and CPUs and each CPU's degree of overutilization.

$$\begin{aligned}\mathcal{T}_{ex} &:= \{\text{Server} \sqsubseteq \exists \text{hasCPU},\ \exists \text{hasCPU}^- \sqsubseteq \text{CPU},\ \text{func}(\text{hasCPU}^-)\}\\ \mathcal{A}_{ex} &:= \{\langle \text{Server}(\text{server}_1), 1\rangle,\ \langle \text{hasCPU}(\text{server}_1, \text{cpu}_1), 1\rangle,\\ &\qquad \langle \text{OverUtilized}(\text{cpu}_1), 0.6\rangle,\ \langle \text{hasCPU}(\text{server}_1, \text{cpu}_2), 1\rangle,\\ &\qquad \langle \text{OverUtilized}(\text{cpu}_2), 0.8\rangle \qquad\qquad\qquad\qquad \}\end{aligned}$$

Based on the ontology $\mathcal{O}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$ we can formulate the following queries:

$$q_1(x) = \text{CPU}(x) \tag{1}$$

$$q_2(x, y) = \text{hasCPU}(x, y) \wedge \text{OverUtilized}(y) \tag{2}$$

$$q_3(x) = \exists y\ \text{hasCPU}(x, y) \wedge \text{OverUtilized}(y) \tag{3}$$

The query q_1 asks for all the CPUs of our system. The query q_2 asks for pairs of Servers and CPUs with an overutilized CPU. The query q_3 asks for Servers for which there exists an overutilized CPU. If conjunction and negation are interpreted based on the Gödel family of operators, the certain answers for each of the queries w.r.t. $\mathcal{O}_{ex}$ are:

$$\begin{aligned} ans(q_1(x), \mathcal{O}_{ex}) &= \{(\text{cpu}_1, 1),\ (\text{cpu}_2, 1)\}\\ ans(q_2(x, y), \mathcal{O}_{ex}) &= \{(\text{server}_1, \text{cpu}_1, 0.6),\ (\text{server}_2, \text{cpu}_2, 0.8)\}\\ ans(q_3(x), \mathcal{O}_{ex}) &= \{(\text{server}_1, 0.8)\}.\end{aligned}$$

3 Fuzzy Reasoning by Extending Crisp Rewritings

Let $q(\mathbf{x})$ be the conjunctive query that the user has formulated over the vocabulary of the $DL\text{-}Lite_R$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$. The main idea underlying the classic $DL\text{-}Lite_R$ reasoning algorithms is to rewrite the query with the information from the TBox and then apply the resulting UCQ to the ABox $\mathcal{A}$ alone. The reasoning algorithm rewrites $q(\mathbf{x})$ by the use of $\mathcal{T}$ into a UCQ $q_{\mathcal{T}}(\mathbf{x})$, called the *rewriting* of q w.r.t. $\mathcal{T}$. For $DL\text{-}Lite_R$-ontologies it is well-known that $\mathcal{O} \models q(\boldsymbol{\alpha})$ iff $\mathcal{A} \models q_{\mathcal{T}}(\boldsymbol{\alpha})$ for any ABox $\mathcal{A}$ and any tuple of individuals in $\mathcal{A}$ holds [2,4]. The PerfectRef$(q, \mathcal{T})$ algorithm, described in [4], computes the rewriting, i.e., the corresponding UCQ.

In order to perform consistency checking for a given $DL\text{-}Lite_R$-ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ the system rewrites the information from $\mathcal{T}$ into a Boolean UCQ $q_{\mathcal{T}}^{\text{unsat}}()$ that contains only existentially quantified variables by the Consistent $(\mathcal{O})$ algorithm, described in [4]. It holds that: an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is inconsistent iff $\mathcal{A} \models q_{\mathcal{T}}^{\text{unsat}}()$.

For fuzzy DLs we adopt the same approach for reasoning. The main difference is that the degrees of ABox assertions must also be taken into account

here. The extensive investigation on the crisp algorithms for *DL-Lite*$_R$ [3,11] and the readily available optimized reasoner systems motivate our investigation on how to employ the classic *DL-Lite*$_R$ algorithm as a black box procedure to perform reasoning for the fuzzy case as well. The main idea is to apply the *DL-Lite*$_R$ rewriting algorithm on the crisp part of the ontology, i.e., by considering assertions as crisp and treating the degrees in a separate form of atoms in a second rewriting step. We apply this idea for satisfiability checking and query answering, extending the classical Consistent($\mathcal{O}$) and PerfectRef($q, \mathcal{T}$)
algorithms to the fuzzy setting.

Before presenting the algorithm, we need introduce some additional notation to accommodate the degrees. For each concept name A we introduce the binary predicate A_f and for each role name P we introduce the ternary predicate P_f. Intuitively, a fuzzy assertion of the form $A(a) \geqslant d$ (or $P(a,b) \geqslant d$) can be represented by a predicate assertion of the form $A_f(a,d)$ (or $P_f(a,b,d)$), where $d \in [0,1]$. The A_f, P_f predicates can be stored as tables in a database. Similarly to the relational database tables tab_A, tab_r of arity 2 and 3 respectively, presented in [12].

Now, to have the fuzzy connectors implemented by the SQL engine correctly, degree variables and degree predicates are needed, which represent the fuzzy operators in the resulting query. These degree variables and predicates are used in the rewritings and enrich the query format used by our algorithms internally. Let $\mathsf{N_{Vd}}$ be a set of *degree variables*. Such degree variables $x_d, y_d \in \mathsf{N_{Vd}}$ can only be mapped to a value in $[0,1]$. By using degree variables in conjunctive queries, we obtain again crisp UCQs with the fuzzy part represented by an additional answer variable x_d.

In order to represent fuzzy conjunction and negation by the t-norm and negation operator described in Table 1, we consider the *degree predicates* $\Phi_>, \Phi_\ominus, \Phi_\otimes$ such that for every $\alpha, \beta, \beta_1, \ldots, \beta_n \in \mathsf{N_{Vd}}$:

$$\Phi_>(\alpha,\beta) = \{(\alpha,\beta) \mid \alpha > \beta\} \tag{4}$$

$$\Phi_\ominus(\alpha,\beta) = \{(\alpha,\beta) \mid \alpha = \ominus\beta\} \tag{5}$$

$$\Phi_\otimes(\alpha,\beta_1,\beta_2) = \{(\alpha,\beta_1,\beta_2) \mid \alpha = \beta_1 \otimes \beta_2\} \tag{6}$$

$$\Phi_\otimes(\alpha,\beta_1,\ldots,\beta_n) = \{(\alpha,\beta_1,\ldots,\beta_n) \mid \alpha = \beta_1 \otimes \ldots \otimes \beta_n\} \tag{7}$$

We call an expression formed over a degree predicate and a tuple of degree variables a *degree atom*. The degree predicates can be materialized in a query language such as SQL or SPARQL by standard mathematical functions and comparison operators. Depending on the family of operators used for fuzzy *DL-Lite*$_R$, the degree predicates $\Phi_\ominus$ and $\Phi_\otimes$ are instantiated according to Table 1.

In the remainder of the paper we use $_f$ to distinguish between the fuzzy and the crisp version of the algorithms and the parameters. For example, the Consistent algorithm used for classic *DL-Lite*$_R$ is extended to the fuzzy case in the Consistent$_f$ algorithm, similarly we use the predicates A and A_f.

3.1 The Consistent$_f$ Algorithm

The Consistent$_f$ method depicted in Algorithm 1 first computes the query $q_{\mathcal{T}}^{\text{unsat}}()$ used for consistency checking in the crisp case. A second rewriting step by RewriteWithDegrees introduces CQs with degree variables and atoms to the query $q_{\mathcal{T}f}^{\text{unsat}}()$ to take into account the degrees from the ABox. The idea is that each CQ in $q_{\mathcal{T}}^{\text{unsat}}()$ corresponds to a different type of inconsistency that may appear in our ontology: line 5 of Function RewriteWithDegrees ensures that no functional restriction is violated, line 7 that no inverse functional restriction is violated, line 9 that no subsumption of the form $A \sqsubseteq_{\mathcal{T}} \neg A'$ is violated, line 11 that no subsumption of the form $A \sqsubseteq_{\mathcal{T}} \neg\exists P$ is violated and so on. Since these are all forms of clashes that can occurr in *DL-Lite*$_R$, the crisp Consistent algorithm produces the UCQ $q_{\mathcal{T}}^{\text{unsat}}()$, which covers all possible cases. The correctness of the method can be shown based on the semantics of fuzzy and crisp *DL-Lite*$_R$.

Example 2. According to the TBox $\mathcal{T} = \{\text{OverUtilized} \sqsubseteq \neg\text{UnderUtilized}\}$ a CPU cannot be in both states of utilization in the crisp case. Therefore, if the conjunctive query $q_{\mathcal{T}_{ex}}^{\text{unsat}}() = \exists x.\text{OverUtilized}(x) \wedge \text{UnderUtilized}(x)$ is entailed, our ontology is inconsistent. However, for the fuzzy case, the degree of OverUtilization should also be taken into account. The query $q_{\mathcal{T}_{ex}}^{\text{unsat}}()$ is rewritten to:

$$q_{\mathcal{T}_{ex}f}^{\text{unsat}}() = \exists x, y_{d_1}, y_{d_2}, y_{d_3}.\text{OverUtilized}_f(x, y_{d_1}) \wedge \text{UnderUtilized}_f(x, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2}).$$

This query asks, if there exists a CPU such that its degree of over-utilization is greater than the negation of its degree of under-utilization. In such a case an entailment $\mathcal{O} \models q_{\mathcal{T}_{ex}f}^{\text{unsat}}()$ would only be given, if $\mathcal{O}$ is inconsistent.

3.2 The PerfectRef$_f$ Algorithm for Answering Conjunctive Queries

Suppose, the conjunctive query $q(\mathbf{x}) = \exists\mathbf{y}.\phi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of concept and role atoms containing variables from $\mathbf{x}, \mathbf{y}$, is to be answered. Based on the crisp *DL-Lite*$_R$ PerfectRef algorithm, the CQ $q(\mathbf{x})$ is rewritten to the $q_{\mathcal{T}}(\mathbf{x})$. This UCQ $q_{\mathcal{T}}(\mathbf{x})$ contains atoms of the form $A(t_1)$ and $P(t_1, t_2)$, where t_1, t_2 are variables in $\mathbf{x}, \mathbf{y}$ or individuals from $\mathcal{O}$. For each CQ $q'(\mathbf{x})$ in the UCQ $q_{\mathcal{T}}(\mathbf{x})$, each atom $A(t_1)$, $P(t_1, t_2)$ is replaced by $A_f(t_1, y_{d'})$, $P_f(t_1, t_2, y_{d'})$ respectively, where $y_{d'}$ is a new degree variable. Likewise, the t-norms of all the degree variables $y_{d'}$ appearing in $A_f(t_1, y_{d'})$ and $P_f(t_1, t_2, y_{d'})$ are added in the extended rewriting in form of degree predicates $\Phi_{\otimes}$. The actual computation of the degree values takes place, when the query is evaluated over the ABox. This idea is made precise in Algorithm 2. This algorithm returns a UCQ that, if answered w.r.t. the ABox $\mathcal{A}$, results in tuples of individuals, along with the degree by which they satisfy the query. If the same tuple of individuals is returned as an answer, but with a different degree, then only the answer with the highest degree is kept.

Algorithm 1. The CONSISTENT$_f$ algorithm

function CONSISTENT$_f$($\mathcal{O}$)
▷ $\mathcal{O}$ is a fuzzy *DL-Lite*$_{\mathcal{RA}}$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$.
  $q_{\mathcal{T}}^{\text{unsat}}() :=$ CONSISTENT(remove-degrees($\mathcal{O}$))
▷ The query $q_{\mathcal{T}}^{\text{unsat}}()$ is obtained from the crisp CONSISTENT algorithm.
  if ans(REWRITEWITHDEGREES($q_{\mathcal{T}}^{\text{unsat}}()$), $\mathcal{A}$) $= \emptyset$ **then**
    return true
  else
    return false
  end if
end function

function REWRITEWITHDEGREES($q_{\mathcal{T}}^{\text{unsat}}()$)
  $q_{\mathcal{T}f}^{\text{unsat}}() := \emptyset$
▷ $q_{\mathcal{T}f}^{\text{unsat}}()$ is an initially empty crisp UCQ.
  for all CQs q in $q_{\mathcal{T}}^{\text{unsat}}()$ **do**
    if q has the form $\exists x, y_1, y_2.P(x, y_1) \wedge P(x, y_2) \wedge y_1 \neq y_2$ **then**
      $q_f := \exists x, y_1, y_2, y_{d_1}, y_{d_2}.P_f(x, y_1, y_{d_1}) \wedge P_f(x, y_2, y_{d_2}) \wedge y_1 \neq y_2 \wedge \Phi_{>}(y_{d_1}, 0) \wedge \Phi_{>}(y_{d_2}, 0)$
▷ q_f the extension of q for querying fuzzy ABoxes.
    else if q has the form $\exists x_1, x_2, y.P(x_1, y) \wedge P(x_2, y) \wedge x_1 \neq x_2$ **then**
      $q_f := \exists x_1, x_2, y, y_{d_1}, y_{d_2}.P_f(x_1, y, y_{d_1}) \wedge P_f(x_2, y, y_{d_2}) \wedge x_1 \neq x_2 \wedge \Phi_{>}(y_{d_1}, 0) \wedge \Phi_{>}(y_{d_2}, 0)$
    else if q has the form $\exists x.A(x) \wedge A'(x)$ **then**
      $q_f := \exists x, y_{d_1}, y_{d_2}, y_{d_3}.A_f(x, y_{d_1}) \wedge A'_f(x, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    else if q has the form $\exists x, y.A(x) \wedge P(x, y)$ **then**
      $q_f := \exists x, y, y_{d_1}, y_{d_2}, y_{d_3}.A_f(x, y_{d_1}) \wedge P_f(x, y, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    else if q has the form $\exists x, y.A(x) \wedge P(y, x)$ **then**
      $q_f := \exists x, y, y_{d_1}, y_{d_2}, y_{d_3}.A_f(x, y_{d_1}) \wedge P_f(y, x, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    else if q has the form $\exists x, y_1, y_2.P(x, y_1) \wedge P'(x, y_2)$ **then**
      $q_f := \exists x, y_1, y_2, y_{d_1}, y_{d_2}, y_{d_3}.P_f(x, y_1, y_{d_1}) \wedge P'_f(x, y_2, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    else if q has the form $\exists x, y_1, y_2.P(x, y_1) \wedge P'(y_2, x)$ **then**
      $q_f := \exists x, y_1, y_2, y_{d_1}, y_{d_2}, y_{d_3}.P_f(x, y_1, y_{d_1}) \wedge P'_f(y_2, x, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    else if q has the form $\exists x, y_1, y_2.P(y_1, x) \wedge P'(y_2, x)$ **then**
      $q_f := \exists x, y_1, y_2, y_{d_1}, y_{d_2}, y_{d_3}.P_f(y_1, x, y_{d_1}) \wedge P'_f(y_2, x, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    else if q has the form $\exists x, y.P(x, y) \wedge P'(x, y)$ **then**
      $q_f := \exists x, y, y_{d_1}, y_{d_2}, y_{d_3}.P_f(x, y, y_{d_1}) \wedge P'_f(x, y, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    else if q has the form $\exists x, y.P(x, y) \wedge P'(y, x)$ **then**
      $q_f := \exists x, y, y_{d_1}, y_{d_2}, y_{d_3}.P_f(x, y, y_{d_1}) \wedge P'_f(y, x, y_{d_2}) \wedge \Phi_{>}(y_{d_1}, y_{d_3}) \wedge \Phi_{\ominus}(y_{d_3}, y_{d_2})$
    end if
    $q_f^{\text{unsat}} := q_f^{\text{unsat}} \cup \{q_f\}$
  end for
return q_f^{unsat}
end function

Algorithm 2. The PERFECTREF$_f$ algorithm

function PERFECTREF$_f(q(\mathbf{x}), \mathcal{T})$
  $q_{\mathcal{T}}(\mathbf{x}) :=$ PERFECTREF$(q(\mathbf{x}), \mathcal{T})$
  $q^f_{\mathcal{T}}(\mathbf{x}) := \emptyset$
  for all CQs $q'(\mathbf{x}) = \exists \mathbf{y}.\phi(\mathbf{x}, \mathbf{y})$ in $q_{\mathcal{T}}(\mathbf{x})$ **do**
    $\mathbf{y_d} := ()$
  ▷ $\mathbf{y_d}$ is a vector that keeps the existentially quantified degree variables.
    $\phi_f(\mathbf{x}, \mathbf{y}) := \emptyset$
  ▷ $\phi_f(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms corresponding to the fuzzy version of $\phi(\mathbf{x}, \mathbf{y})$.
    for all $A(t)$ in $q'(\mathbf{x})$ **do**
      Add the degree variable $y_{d'}$ to the vector $\mathbf{y_d}$
  ▷ $y_{d'}$ is a fresh degree variable name.
      $\phi_f(\mathbf{x}, \mathbf{y}) := \phi_f(\mathbf{x}, \mathbf{y}) \wedge A_f(t, y_{d'})$
    end for
    for all $P(t_1, t_2)$ in $q'(\mathbf{x})$ **do**
      Add the degree variable $y_{d'}$ to the vector $\mathbf{y_d}$
      $\phi_f(\mathbf{x}, \mathbf{y}) := \phi_f(\mathbf{x}, \mathbf{y}) \wedge P_f(t_1, t_2, y_{d'})$
    end for
    $q'_f(\mathbf{x}, x_d) := \exists \mathbf{y}, \mathbf{y_d}.\phi_f(\mathbf{x}, \mathbf{y}) \wedge \Phi_{\otimes}(x_d, \mathbf{y_d})$
    $q^f_{\mathcal{T}}(\mathbf{x}, x_d) := q^f_{\mathcal{T}}(\mathbf{x}, x_d) \cup \{q'_f(\mathbf{x}, x_d)\}$
  end for
  return $q^f_{\mathcal{T}}(\mathbf{x})$
end function

Example 3. Based on $\mathcal{O}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$ from Example 1 we illustrate the application of PERFECTREF$_f$ algorithm to the queries q_1, q_2, q_3 from Example 1. Initially, q_1, q_2, q_3 are rewritten, by the crisp PERFECTREF algorithm to the following UCQs:

$$\begin{aligned} q_{1\,\mathcal{T}_{ex}}(x) =& \{\mathrm{CPU}(x), \quad \exists y.\mathrm{hasCPU}(y, x)\} \\ q_{2\,\mathcal{T}_{ex}}(x, y) =& \{\mathrm{hasCPU}(x, y) \wedge \mathrm{OverUtilized}(y)\} \\ q_{3\,\mathcal{T}_{ex}}(x) =& \{\exists y.\mathrm{hasCPU}(x, y) \wedge \mathrm{OverUtilized}(y)\} \end{aligned}$$

In the next step, the PERFECTREF$_f$ algorithm extends the queries with degree variables and atoms, so that the corresponding degrees can be returned:

$$\begin{aligned} q_1{}^f_{\mathcal{T}_{ex}}(x, x_d) =& \{\mathrm{CPU}(x, x_d), \quad \exists y.\mathrm{hasCPU}(y, x, x_d)\} \\ q_2{}^f_{\mathcal{T}_{ex}}(x, y, x_d) =& \{\mathrm{hasCPU}(x, y, y_{d_1}) \wedge \mathrm{OverUtilized}(y, y_{d_2}) \wedge \Phi_{\otimes}(x_d, y_{d_1}, y_{d_2})\} \\ q_3{}^f_{\mathcal{T}_{ex}}(x, x_d) =& \{\exists y.\mathrm{hasCPU}(x, y, y_{d_1}) \wedge \mathrm{OverUtilized}(y, y_{d_2}) \wedge \Phi_{\otimes}(x_d, y_{d_1}, y_{d_2})\} \end{aligned}$$

For the ABox $\mathcal{A}_{ex}$ we get the following set of answers to each of the queries:

$$\begin{aligned} ans(q_1{}^f_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex}) =& \{(cpu_1, 1), (cpu_2, 1)\} \\ ans(q_2{}^f_{\mathcal{T}_{ex}}(x, y, x_d), \mathcal{A}_{ex}) =& \{(server_1, cpu_1, 0.6), (server_1, cpu_2, 0.8)\} \\ ans(q_3{}^f_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex}) =& \{(server_1, 0.6), (server_1, 0.8)\} \end{aligned}$$

Finally, for each answer to a query, only the one with the highest degree is kept per (tuple of) individual(s):

$$ans(q_1{}^{f}_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex}) = \{(cpu_1, 1), (cpu_2, 1)\}$$
$$ans(q_2{}^{f}_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex}) = \{(server_1, cpu_1, 0.6), (server_1, cpu_2, 0.8)\}$$
$$ans(q_3{}^{f}_{\mathcal{T}_{ex}}(x, x_d), \mathcal{A}_{ex}) = \{(server_1, 0.8)\}$$

Unfortunately, this practical approach does not always yield correct results. The simplifications made during the rewriting step by the crisp algorithms PerfectRef and Consistent are correct for the crisp, but not for the fuzzy case. Specifically, a conjunctive query that contains the atom $A(x)$ repeatedly is simplified in the crisp case to a conjunctive query containing the same atom only once—an obvious optimization for the crisp case. However, in the fuzzy case, such simplification causes our algorithm to become unsound, since for every $A^{\mathcal{I}}(o) \in (0, 1)$ it applies that $A^{\mathcal{I}}(o) > A^{\mathcal{I}}(o) \otimes A^{\mathcal{I}}(o)$ for the Łukasiewicz and product families of operators. Similarly, each time two atoms are unified during the rewriting, one contribution degree is lost. These effects are better illustrated by the following example.

Example 4. Suppose that $\otimes$ is the product ($\times$) t-norm and our ontology has the following TBox and ABox:

$$\mathcal{T} = \{A_1 \sqsubseteq A_2,\ A_3 \sqsubseteq A_4\} \qquad \mathcal{A} = \{A_1(a) \geqslant 0.8,\ A_3(a) \geqslant 0.9\}.$$

Then the conjunctive query $q(x) = A_1(x) \wedge A_2(x) \wedge A_3(x) \wedge A_4(x)$ has a as an answer with degree ≥ 0.5184, since ${A_1}^{\mathcal{I}}(a^{\mathcal{I}}) \times {A_1}^{\mathcal{I}}(a^{\mathcal{I}}) \times A_3^{\mathcal{I}}(a^{\mathcal{I}}) \times A_3^{\mathcal{I}}(a^{\mathcal{I}}) = 0.5184$. Now, the crisp algorithm returns the following UCQ as rewriting:

$$q_{\mathcal{T}}(x) = \{A_1(x) \wedge A_2(x) \wedge A_3(x) \wedge A_4(x),\ A_1(x) \wedge A_3(x) \wedge A_4(x),$$
$$A_1(x) \wedge A_2(x) \wedge A_3(x),\ A_1(x) \wedge A_3(x)\}$$

For the crisp case there is no difference between the answers to the conjunctive queries $A_1(x) \wedge A_3(x)$ or $A_1(x) \wedge A_1(x) \wedge A_3(x) \wedge A_3(x)$. If we apply our rewriting technique for fuzzy queries to the last query, we get a fuzzy conjunctive query of the form:

$$q^{f}_{\mathcal{T}}(x, x_d) = \exists y_{d_{A_1}}, y_{d_{A_3}}.A_{1f}(x, y_{d_{A_1}}) \wedge A_{3f}(x, y_{d_{A_3}}) \wedge \Phi_{\times}(x_d, y_{d_{A_1}}, y_{d_{A_3}}) \quad (8)$$

and the answer for the variables x and x_d is $(a, 0.72)$, i.e., a is an answer with a degree ≥ 0.72 instead of 0.5184 which is the correct degree.

To conclude, our pragmatic approach for query answering over a fuzzy ontology, that uses the rewritings obtained during crisp query answering, yields sound results fuzzy semantics with idempotent operators such as the Gödel family of operators. For other families of operators, that are not idempotent, the algorithm need not be sound in the sense that the degree of a result returned may be greater than the actual degree.

4 Limitations of the Approach

4.1 Identifying and Assessing Unsound Results for Non-idempotent Fuzzy DLs

Since our approach for conjunctive query answering is sound for the Gödel family of operators, a natural question is when a case that might yield an unsound result is encountered. To this end we present a straightforward idea for identifing unsound results for the degrees and to give a narrowed down interval for the missed degrees. Recall that the *DL-Lite*$_R$-CQs have concept or role atoms, whereas the UCQs returned from our algorithms have degree atoms in addition. Let $|q(\mathbf{x})|_{\mathsf{CR}}$ denote the number of concept and role atoms of a CQ (degree atoms are not taken into account), and let $q^f_{\mathcal{T}}(\mathbf{x})$ be the UCQ that the algorithm PerfectRef$_f$ returns. A property of the crisp *DL-Lite*$_R$ algorithm is that

$$|q(\mathbf{x})|_{\mathsf{CR}} \geqslant |q'_f(\mathbf{x}, x_d)|_{\mathsf{CR}} \quad \text{for every } q'_f(\mathbf{x}, x_d) \in q^f_{\mathcal{T}}(\mathbf{x}).$$

This property allows to infer: if $|q(\mathbf{x})|_{\mathsf{CR}} = |q'_f(\mathbf{x}, x_d)|_{\mathsf{CR}}$ for every CQ $q'_f(\mathbf{x}, x_d)$ in $q^f_{\mathcal{T}}(\mathbf{x})$, then no atom simplification has been applied and thus our algorithm gave a correct result.

In case $|q(\mathbf{x})|_{\mathsf{CR}} \geqslant |q'_f(\mathbf{x}, x_d)|_{\mathsf{CR}}$ for some $q'_f(\mathbf{x}, x_d) \in q^f_{\mathcal{T}}(\mathbf{x})$, a pessimistic estimation for the not correctly calculated degrees can be computed in the following way. Suppose that $|q(\mathbf{x})|_{\mathsf{CR}} = n$, while $|q'_f(\mathbf{x}, x_d)|_{\mathsf{CR}} = m$ with $n > m$. Based on the PerfectRef algorithm, each concept and role atom in $q(\mathbf{x})$ can be mapped to some corresponding 'fuzzy' atom in $q'_f(\mathbf{x}, x_d)$. Since $n > m$, there is at least one atom in $q'_f(\mathbf{x}, x_d)$ to which several atoms in $q(\mathbf{x})$ map to. Thus a simplification has taken place and the degree variables of some of the atoms in $q'_f(\mathbf{x}, x_d)$ are not calculated correctly. In fact, exactly $n-m$ occurrences of degree variables are ignored. Since $|q'_f(\mathbf{x}, x_d)|_{\mathsf{CR}} = m$, the query $q'_f(\mathbf{x}, x_d)$ contains the predicate $\Phi_{\otimes}(x_d, y_{d1}, \ldots, y_{dm})$, where $x_d, y_{d1}, \ldots, y_{dm} \in \mathsf{N}_{\mathsf{Vd}}$. Each such simplification step causes a predicate atom $A_f(t_i, y_{di})$ or $P_f(t_i, t'_i, y_{di})$ with $1 \leq i \leq n$ occurring in $q(\mathbf{x})$ being omitted when computing the membership degree for the conjunction in $q'_f(\mathbf{x}, x_d)$ by evaluating $\Phi_{\otimes}(x_d, y_{d1}, \ldots, y_{dm})$. Since it is unknown which of the predicate atoms and consequently which degree variable is missing, we consider the most pessimistic case, i.e., that the variable taking the lowest degree in each answer has not been calculated. This minimum value is represented in the variable y_λ: $\Phi_{\min}(y_\lambda, y_{d1}, \ldots, y_{dm})$ (the predicate $\Phi_{\min}$ corresponds to the predicate $\Phi_{\otimes}$ in Equation 7 where $\otimes$ is replaced by the min t-norm). The membership degree for the pessimistic case can be calculated by changing the line 15 of Algorithm 2, so that the value of the degree variable x_d in the query is calculated by:

$$\Phi_{\otimes}(x_d, \mathbf{y_d}, \underbrace{y_\lambda, \ldots, y_\lambda}_{n-m \text{ times}}) \wedge \Phi_{\min}(y_\lambda, y_{d1}, \ldots, y_{dm}).$$

The difference of the value returned by the algorithm and the value from the pessimistic estimation, give an estimate how close the returned answer is to the correct answer.

Example 5. Extending Example 4, the query to acquire a pessimistic degree estimation is:

$$q^f_{\mathcal{T}}(x, x_d) = \exists y_{d_A}, y_{d_B}, y_\lambda . A_f(x, y_{d_A}) \wedge B_f(x, y_{d_B}) \wedge \\ \wedge \Phi_\times(x_d, y_{d_A}, y_{d_B}, y_\lambda, y_\lambda) \wedge \Phi_{\min}(y_\lambda, y_{d_A}, y_{d_B}).$$

In the single pessimistic answer returned, y_λ takes the value of 0.8 and the estimation is that a is an answer to the query with a degree ≥ 0.4608. This estimation is very close to the correct one, i.e., a is the answer to the query with a degree ≥ 0.5184. Now, with the pessimistic answer $(a, 0.4608)$ and the unsound answer $(a, 0.72)$, we know that the correct degree is between the two values.

4.2 Extended Use of Fuzzy Information

Our pragmatic approach can only handle fuzzy information in ABox assertions. Sometimes it can be useful to have also concept inclusion axioms with degrees or to extend conjunctive queries by fuzzy information.

Fuzzy *DL-Lite*$_R$ with Degrees in Concept Inclusions: So far we have only considered concept inclusions of the form $B \sqsubseteq C$ in the extended rewriting approach. To extend our approach to the general case of fuzzy concept inclusions, i.e., $\langle B \sqsubseteq C, d\rangle$, is not straightforward. Such concept inclusions are satisfied by an interpretation $\mathcal{I}$ iff for every $\delta \in \Delta^{\mathcal{I}}$ and the implication operator from Table 1:

$$(B^{\mathcal{I}}(\delta) \Rightarrow C^{\mathcal{I}}(\delta)) \geqslant d.$$

We present here the intuition what the obstacles are. Suppose that our algorithm contains the concept inclusion $\langle B \sqsubseteq C, d\rangle$ and the corresponding CQ contains only the atom $C(x)$. During the rewriting, the replacement of $C_f(x, y_C)$ by $B_f(x, y_B)$ takes place and the degree d should also to be calculated, i.e., the CQ returned after the replacement should be $\exists y_B . B_f(x, y_B) \wedge \Phi_\otimes(x_d, y_B, d)$, where d is a degree and not a degree variable. Unfortunately, this cannot be done by the crisp rewriting algorithm since it does not keep track of the degrees in fuzzy concept inclusions.

One could introduce a new set of concept names corresponding to the α-cuts of each concept, similar to the reduction technique presented in [10]. Here, the concept $B_{\geqslant 0.3}$ represents the set of elements that belong to the concept B with a degree greater or equal than 0.3 and the concept inclusion $\langle B \sqsubseteq C, d\rangle$ can be replaced by the set of concept inclusions: $\langle B_{\geqslant d \otimes d'} \sqsubseteq C_{\geqslant d'}, 1\rangle$ for each degree d' in $\mathcal{T}$. Then in the final query each concept atom $B_{f\,\geqslant d}(x, y_{d_B})$ is replaced by $B_f(x, y_{d_B})$ and the degree d is simply used in the predicate atom $\Phi_\otimes(\dots)$.

This procedure would remedy the above problem, but it would not yield optimized queries for the following reasons:

– Simplifications, optimizations and variable unifications are not performed since the crisp DL-Lite algorithm lacks the information that $B_{\geqslant 0.3}$ and $B_{\geqslant 0.4}$ are different α-cuts of the same concept.

– If there are n nested replacements in the rewriting, then the algorithm would need to compute all possible products of n factors for the Łukasiewicz and product families of operators.

Therefore this method needs to be further investigated regarding its applicability and effectiveness.

Fuzzy *DL-Lite*$_R$ with Generalized Query Component: A generalized form of fuzzy CQs are those queries in which a score of a query is computed via a monotone scoring function. Such kind of queries have already been investigated in [6,12] and the question is whether our black box approach can be applied to answer them as well. Extending Example 4, we can express via a scoring function that the parameter A_3 is more important than A_4 which in turn is more important than A_1 and A_2:

$$q(x) = 0.2 \cdot A_1(x) + 0.1 \cdot A_2(x) \wedge 0.4 \cdot A_3(x) + 0.3 \cdot A_4(x). \tag{9}$$

Again, due to the simplifications taking place in the crisp rewriting step, some of the atoms may be merged and therefore after this step the initial weight corresponding to the merged atoms are unknown. For equation 9, the crisp PerfectRef algorithm returns an UCQ containing, among others, the CQ $A_1(x) \wedge A_3(x)$. For this CQ, one cannot guess correctly how to assign the weights $0.2, 0.1, 0.4, 0.3$ to the two remaining atoms.

Fuzzy *DL-Lite*$_R$ Threshold Queries: Another interesting form of queries w.r.t. to a fuzzy ontology, are threshold queries. These queries ask for all individuals that satisfy each atom with at least a certain degree. Threshold conjunctive queries may take the following form:

$$q(x) = \mathsf{Server(x)} \geqslant 1 \wedge \mathsf{hasPart}(x, y) \geqslant 1 \wedge \mathsf{CPU}(y) \geqslant 1 \wedge \mathsf{Overutilized}(y) \geqslant 0.4$$

Again, due to the simplifications taking place, threshold queries cannot be handled directly by employing the crisp rewritings first.

5 Practical Implementation and Performance Test

5.1 The FLite Reasoner

We have developed a reasoner for conjunctive query answering with respect to a TBox $\mathcal{T}$ and a fuzzy ABox $\mathcal{A}$ for *DL-Lite*$_R$. FLite (Fuzzy *DL-Lite*$_R$ query engine) implements the query answering algorithm presented in Section 3 and it builds on the rewriting algorithms for crisp *DL-Lite*$_R$ implemented in the Ontop framework [8] developed at the Free University of Bozen Bolzano.

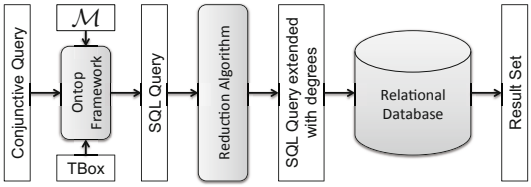

Fig. 1. FLITE implementation

Figure 1 illustrates the whole query answering pipeline and the components involved. The initial input is a conjunctive query $q(\mathbf{x})$ represented in the form of a SPARQL query. The Ontop framework requires that the ABox $\mathcal{A}$ is stored in a relational database. A mapping $\mathcal{M}$, in the form of multiple SQL queries, translates the Tables of the relational database to ABox assertions. By combining the mapping $\mathcal{M}$ with the TBox assertions, the Ontop framework rewrites the initial query to a UCQ $q_{\mathcal{T}}(\mathbf{x})$, in the form of an SQL query. The rewritten query is post-processed by FLITE, as described in Section 3, resulting in the UCQ $q^f_{\mathcal{T}}(\mathbf{x}, x_d)$ that additionally asks for the associated degree of each answer by means of degree variables. The final SQL query is then evaluated over the relational database returning the corresponding result set with degrees.

Example 6. Let's consider again the three conjunctive queries and the ontology $\mathcal{O}_{ex}$ from Example 1 and assume that ABox $\mathcal{A}_{ex}$ is stored in a relational database. The mapping $\mathcal{M}$ is used to map the set of answers to: i) the query `select Server_id from Servers` to instances of the concept *Server*, ii) the query `select Server_id,CPU_id from CPUs` to instances of the role *hasCPU*, iii) the query `select CPU_id,Degree from Overutilized` to instances of the concept OverUtilized along with their corresponding degree. In this example, only the concept OverUtilized is fuzzy. It is represented by rows in the Table Overutilized stating the *CPU* and its degree of over-utilization. For an entry (cpu_1, 0.6) in Table Overutilized we have that OverUtilized(cpu_1) $\geqslant 0.6$, all the other concepts are crisp and therefore have a degree of 1.0. Next the Ontop framework transforms the CQ in equation 2 to the following SQL query (in black), which is augmented by our extended rewriting algorithm (in gray).

```
SELECT    QVIEW1.Server_id AS x, QVIEW1.CPU_id AS y,
QVIEW2.Degree AS d
FROM  CPUs QVIEW1,Overutilized QVIEW2
WHERE     QVIEW1.Server_id IS NOT NULL AND QVIEW1.
    CPU_id IS NOT NULL AND (QVIEW1.CPU_id = QVIEW2.
    CPU_id)
```

5.2 An Initial Performance Evaluation

We have evaluated the performance of FLITE on an ontology. The current version of the HAEC fuzzy *DL-Lite*$_R$ ontology contains 311 TBox axioms, 178 concepts, 39 roles, together with 15 conjunctive queries. We performed our evaluation for a complicated query containing 13 concept and role atoms. Out of these 13 atoms, 9 were about fuzzy concepts, thus the extended SQL contained 9 additional degree variables. Out of the 10 relational database tables used to store the fuzzy ABox information 4 contained fuzzy information. Thus, about

40% of the ABox assertions were fuzzy. We evaluated the performance of our approach by comparing FLite to the standard Ontop framework for the classic $DL\text{-}Lite_R$ language by simply ignoring the degrees in concept assertions.[1]

The evaluation of the system was performed on a MacBook Pro laptop with *2.6 GHz Intel Core i7 Processor*, *8 GB 1600 MHz DDR3 Memory*, running a *PostgreSQL 9.3.4 on x86_64-apple-darwin* database. Figure 2 depicts the comparison between Ontop and its extension FLite in terms of running time. The graph shows the performance of the two query engines w.r.t. the number of assertions in the ABox. As we can see the overhead of adding degrees and answering queries containing degrees can be handled well by our algorithm and the database. In fact, FLite answered the queries having to examine up to 326,340 ABox assertions within only 1,519 *ms* for the crisp and within 2,717 *ms* for the fuzzy case.

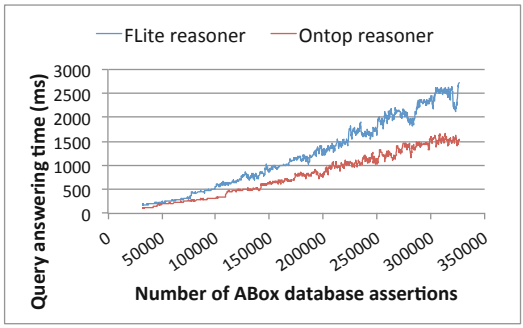

Fig. 2. Running times: Ontop - FLite

6 Conclusions

We presented a pragmatic approach for answering conjunctive queries over ontologies with fuzzy ABoxes. Our approach uses rewritings obtained by the algorithm for answering crisp queries. Although described here for $DL\text{-}Lite_R$, our approach can be extended to other DLs that enjoy FOL rewritability. Our algorithm is sound for those t-norms that have idempotent operators, such as the Gödel t-norm. This does not need be for other t-norms. We devised a method by which unsound answers can be identified and the correct degrees estimated. We implemented our approach in the FLite system and evaluated it against the Ontop framework. Our initial experiments suggest that the overhead for handling fuzzy information does not crucially affect the overall performance.

Our extended rewriting approach cannot be extended in straight-forward way to other interesting forms of queries such as threshold queries. To answer these kind of queries one would have to implement an algorithm from scratch [6,13] or extend the source code of an existing rewriting implementation. A thorough investigation of this subject remains future work.

[1] A comparison of the performance of FLitewith SoftFacts [13] –an ontology mediated database system based on the DLR-Lite language–would have been more appropriate, but the system could not be set up.

References

1. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: Quonto: querying ontologies. In: AAAI, pp. 1670–1671 (2005)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-lite family and relations. Journal of Artificial Intelligence Research **36**(1), 1–69 (2009)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and databases: the *DL-Lite* approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-lite family. Journal of Automated Reasoning **39** (2007)
5. Mailis, T., Peñaloza, R., Turhan, A.-Y.: Conjunctive query answering in finitely-valued fuzzy description logics. In: Kontchakov, R., Mugnier, M.-L. (eds.) RR 2014. LNCS, vol. 8741, pp. 124–139. Springer, Heidelberg (2014)
6. Pan, J.Z., Stamou, G.B., Stoilos, G., Thomas, E.: Expressive querying over fuzzy DL-Lite ontologies. In: Description Logics (2007)
7. Poggi, A., Rodriguez, M., Ruzzi, M.: Ontology-based database access with DIG-Mastro and the OBDA plugin for protégé. In: Proc. of OWLED (2008)
8. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: *Ontop* of databases. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 558–573. Springer, Heidelberg (2013)
9. Stocker, M., Smith, M.: Owlgres: a scalable OWL reasoner. In: OWLED, vol. 432 (2008)
10. Straccia, U.: Transforming fuzzy description logics into classical description logics. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 385–399. Springer, Heidelberg (2004)
11. Straccia, U.: Answering vague queries in fuzzy DL-Lite. In: Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, (IPMU 2006), pp. 2238–2245 (2006)
12. Straccia, U.: Towards top-k query answering in description logics: the case of DL-Lite. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 439–451. Springer, Heidelberg (2006)
13. Straccia, U.: Softfacts: a top-k retrieval engine for ontology mediated access to relational databases. In: 2010 IEEE International Conference on Systems Man and Cybernetics (SMC), pp. 4115–4122. IEEE (2010)
14. Venetis, T., Stoilos, G., Stamou, G.: Query extensions and incremental query rewriting for OWL 2 QL ontologies. Journal on Data Semantics pp. 1–23 (2014)